УДК 004.4

Исследование методов и средств проектирования отказоустойчивых и высоконадежных решений для облачных баз данных их реализация как надстройки для базы данных PostgreSQL

Бакин А.О. (Университет ИТМО)
Научный руководитель – доцент, кандидат технических наук, Перл И.А.
(Университет ИТМО)

Введение. Автоматизация развертывания и использования СУБД сопровождается рядом трудностей, ввиду требований, предъявляемых бизнесом. Во-первых, СУБД должна быть высоконадежной и высокодоступной. Во-вторых, может потребоваться развертывание дополнительных сервисов, например connection pool. В-третьих, необходимо автоматизировать создание резервных копий. Также, могут предъявляться и дополнительные требования: статичность IP адреса, GUI настройки, cloud-native и т.д.

Все СУБД имеют уникальную архитектуру, а значит, к каждой из них требуется индивидуальный подход для реализации описанных выше требований, а проблема их реализации более серьезно стоит для открытых СУБД. В качестве целевой СУБД в данной работе была выбрана одна из самых популярных и открытых – PostgreSQL. Также стоит отметить, что растет потребность в развертывании СУБД с использованием облачных технологий, поэтому решения можно поделить на два типа: не облачные и облачные.

К не облачным можно отнести классическое решение на базе Patroni. Данное решение отлично реализует только требование надежности (первое), однако, для реализации остальных требуется ручное развертывание еще нескольких сервисов. Это не только усложняет процесс, но и уменьшает надежность всей системы в целом. Также, для интеграции с облачными сервисами требуется дополнительная работа инженеров. Похожие проблемы присущи и другим, менее популярным не облачным решениям.

К облачным можно отнести Kubernetes операторы вроде Crunchy Data или Zalando. Они основаны на Patroni и позволяют серьезно упростить работу инженеров по развертыванию сервисов. Однако, во-первых эти операторы работают только в среде Kubernetes, а ни у каждой компании есть возможность ее использования. Во-вторых же, растет ненадежность системы ввиду использования множества сервисов “поверх” PostgreSQL. Стоит отметить, что для Docker задача автоматизации развертывания PostgreSQL решается ручной адаптацией не облачного решения вроде Patroni, так как готовые решения отсутствуют.

Существуют также и сервисы от облачных провайдеров вроде Azure, AWS и д.р. Однако, такой вариант подходит не всем по ряду причин: цена, недоступность в регионе/стране, безопасность чувствительных данных.

Основная часть. Для решения обозначенных проблем была разработана архитектура решения PgFacade. Упор делается на использование облачных технологий так как они идеально подходят для автоматизации развертывания и работы СУБД, а их популярность при этом растет. В основе лежит оркестратор, управляющий PostgreSQL, который должен уметь работать в любой облачной среде (Docker, Kubernetes и др.). При этом, решение должно поддерживать концепцию “Plug and Play” для минимизации труда разработчиков. Также, решение должно быть единым сервисом, то есть, совмещать в себе функции: прокси, connection pool, оркестратор и т.д. Это обеспечит надежность.

Было решено разделить всю систему на 3 слоя:

1. Слой, содержащий узлы СУБД PostgreSQL, оркестрируемые слоем 2. В данном слое всегда должен быть только один мастер, а также задаваемое число реплик. Мастер поддерживает операции чтения и записи, а реплики только чтение. В случае

сбоя мастера, оркестратор заменит его одной из реплик. Таким образом достигается отказоустойчивость и высокая доступность.

2. Слой, содержащий узлы приложения PgFacade. Помимо оркестровки PostgreSQL, слой также является единым сервисом, обеспечивающим почти все остальные требования. Для того, чтобы данный слой не стал “узким местом”, он содержит несколько узлов, взаимодействующих на основе алгоритма Raft. Только узел с ролью “лидер” занимается оркестровкой, а узлы с ролью “фолловер” могут только проксировать запросы. В случае недоступности лидера, благодаря алгоритму Raft, будет избран новый лидер.
3. Опциональный слой, содержащий балансировщик нагрузки. В его задачи входит обеспечение статичности IP адреса и балансировки запросов на подключение между узлами слоя 2. Также, балансировщик может быть улучшен, чтобы не “обрывать” соединения конечных пользователей даже в случае выхода из строя узла из слоя 2.

На основе предложенной архитектуры было разработано приложение, а также адаптер для его работы в среде Docker. Для подтверждения применимости решения было проведено тестирование, при котором узлы системы отключались случайным образом, после чего фиксировалось восстановление системы. Также, было протестировано влияние решения на задержку ответа от СУБД, показавшее незначительное влияние.

Выводы. Были проанализированы существующие решения для автоматизации развертывания и управления СУБД PostgreSQL и показаны их недостатки. Для устранения недостатков была разработана архитектура оркестратора PostgreSQL в облачной среде. Благодаря тестированию было проверено, что решение реализует описанные требования, при этом не становясь “узким местом” и не сильно влияя на время отклика СУБД. Простота использования делают решение PgFacade выигрышным, на фоне существующих решений.

Список использованных источников:

1. Официальная документация PostgreSQL – URL: https://www.postgresql.org/docs/ (дата обращения: 06.02.2023). – Текст: электронный.
2. Ongaro Diego, Ousterhout John. In Search of an Understandable Consensus Algorithm – URL: https://web.stanford.edu/~ouster/cgi-bin/papers/raft-atc14 (дата обращения: 06.02.2023). – Текст: электронный.
3. Pablo Bárbaro & Martinez Pedroso, High availability and load balancing for PostgreSQL databases: designing and implementing. // International Journal of Database Management Systems (IJDMS) Vol.8, No.6, December 2016